NASA/TM—2000-209950

ARL–MR–488

Developing CORBA-Based Distributed Scientific Applications From Legacy Fortran Programs

Janche Sang
Cleveland State University, Cleveland, Ohio

Chan Kim
Glenn Research Center, Cleveland, Ohio

Isaac Lopez
U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio

July 2000

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/TM—2000-209950 ARL–MR–488

Developing CORBA-Based Distributed Scientific Applications From Legacy Fortran Programs

Janche Sang
Cleveland State University, Cleveland, Ohio

Chan Kim
Glenn Research Center, Cleveland, Ohio

Isaac Lopez
U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio

Prepared for the
Computational Aerosciences Workshop
sponsored by the High Performance Computing and Communications Program
Moffett Field, California, February 15–17, 2000

National Aeronautics and
Space Administration

Glenn Research Center

July 2000

Acknowledgments

The authors express their appreciation to management of the High Performance Computing and Communications Program and to the NASA R&T Base Program for supporting NPSS.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

Available electronically at http://gltrs.grc.nasa.gov/GLTRS/

DEVELOPING CORBA-BASED DISTRIBUTED SCIENTIFIC APPLICATIONS FROM LEGACY FORTRAN PROGRAMS

Janche Sang
Cleveland State University
Cleveland, Ohio 44115

Chan M. Kim
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Isaac Lopez
U.S. Army Research Laboratory
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

SUMMARY

An efficient methodology is presented for integrating legacy applications written in Fortran into a distributed object framework. Issues and strategies regarding the conversion and decomposition of Fortran codes into Common Object Request Broker Architecture (CORBA) objects are discussed. Fortran codes are modified as little as possible as they are decomposed into modules and wrapped as objects. A new conversion tool takes the Fortran application as input and generates the C/C++ header file and Interface Definition Language (IDL) file. In addition, the performance of the client server computing is evaluated.

INTRODUCTION

Recent progress in distributed object technology has enabled software applications to be developed and deployed easily such that objects or components can work together across network boundaries, in different operating systems, and in different languages. A distributed object is not necessarily a complete application but rather a reusable, self-contained piece of software that cooperates with other objects in a plug-and-play fashion via a well-defined interface. The Common Object Request Broker Architecture (CORBA), a middleware standard defined by the Object Management Group (OMG) (ref. 1), uses the Interface Definition Language (IDL) to specify such an interface for transparent communication between distributed objects. Since IDL can be mapped to any programming language, such as C++, Java (Sun Microsystems), or Smalltalk, existing applications can be integrated into a new application and the tasks of rewriting code and maintaining software can be reduced.

In OMG's object model, an object is an encapsulated entity with a distinct immutable identity. Its services can be accessed only through interfaces defined in IDL (ref. 2). Clients issue requests to objects to perform services on their behalf, but the implementation and location of each object are hidden from the requesting client. Communication between clients and objects is provided by the Object Request Broker (ORB), a key component of the CORBA architecture. Upon compiling an IDL file, the ORB generates the stub and the skeleton through which a client can invoke a method on a server object, which can be on the same machine or across a network. The ORB is responsible for finding an object that can implement the request, passing it the parameters, invoking its method, and returning the results to the client. In this process, the client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an objects interface.

Since its inception, CORBA has been widely accepted as the middleware standard for distributed object computing. It relieves distributed application developers of the cumbersome task of dealing with issues due to heterogeneous computing environments, and it provides a standards-based interface to facilitate transparent exchange of management information for computer and communication networks (refs. 3 and 4). TeleMed (ref. 5) is an effort to demonstrate sharing multimedia electronic medical records over a wide area network. It was designed as a distributed object system in which the various healthcare components are dealt with as objects and distributed via the CORBA standard. CORBA-based distributed object technology is considered the key to integrating legacy applications in highly dynamic business environments (ref. 6). Industry-specific success stories about CORBA

applications are reported in OMG's web site (ref. 7). OMG's Domain Technical Committee aims at developing domain-specific CORBA services and technologies in various industries.

Many scientific applications in aerodynamics and solid mechanics are written in Fortran. Refitting these legacy Fortran codes with CORBA objects can increase the code reusability. For example, scientists could link their specific applications to objectified vintage Fortran programs such as Partial Differential Equation solvers, in a plug-and-play fashion. Many standalone Fortran applications developed to analyze the performance of an individual component of the engineering system can be converted to CORBA objects and then combined with other objects to design the entire system. Reference 8 documents attempts to provide a collaborative design and simulation environment based on this concept. A CORBA-based software environment is developed in reference 9. It couples two independently developed codes written in Fortran and C++ to model a thermomechanical problem. A computationally intensive Fortran application also can be decomposed into several pieces, made into CORBA objects, and distributed over several machines to speed up the computation. Unfortunately, CORBA-IDL-to-Fortran mapping has not been proposed, and there seems to be no direct method of generating CORBA objects from Fortran without manually writing C/C++ wrappers.

In this paper, we present an efficient methodology to integrate applications written in Fortran into a distributed object framework. Issues and strategies regarding the conversion and decomposition of Fortran codes into CORBA objects are discussed. Our goal was to keep the Fortran codes unmodified as much as possible. To reduce the programming effort in code wrapping, we designed and implemented a conversion tool that takes the Fortran application program as input and generates the C/C++ header file and IDL file. At the end of this paper, we evaluate the performance of the client-server computing and identify possible communication overhead.

METHODOLOGY

One method of wrapping a legacy application is to encapsulate the entire legacy code into a single object. Programmers only need to provide a server that will invoke the wrapped legacy-code object when it receives a request from a client. This straightforward method is suitable for small-scale applications that have only one module or entity.

For complicated applications, an alternative method that we are interested in is to decompose the codes into different function modules and wrap each module into a distributed object. This method can increase the code usability because each distributed object can be invoked by a different application as a plug-and-play software component. Figure 1 shows the conversion and decomposition mechanism we proposed. Our objective was to keep modifications of the Fortran source codes to a minimum. The conversion tool takes the Fortran application program as input and helps programmers generate a C/C++ header file and an IDL file for wrapping the Fortran code.

In our current environment, individual programmers need to determine how to decompose the legacy application into several reusable components on the basis of the cohesion and coupling factors of the functions and subroutines. In the future, we plan to add an analyzer tool to help programmers extract objects from legacy codes. Earlier studies in object extraction can be found in references 10 and 11. This topic is beyond the scope of this paper.

Most Fortran applications use the COMMON block to distribute a large number of variables among several functions. COMMON blocks play a role similar to that of the global variables used in C. In the CORBA-compliant programming environment, global variables cannot be used to pass values between objects. One approach to dealing with such problem is to put the COMMON variables into the parameter list, but this requires extensive modification of the Fortran source code, which violates our design considerations. Our approach is to extract the COMMON blocks and convert them into a structure-type attribute in C++. Through the attributes, each component can initialize the variables and return the computed result back to the client. With our conversion tool, the programming effort can be greatly reduced because function headings, types, and even the COMMON blocks are converted to C++ and IDL styles.

IMPLEMENTATION ISSUES

The conversion tool we proposed in the previous section consists of a parser and a code generator. The parser constructs parsing trees from the input Fortran codes, and the generator translates the trees into C++/IDL codes. Instead of writing a language translator from the beginning, we implemented the conversion tool with an existing Fortran-to-C converter called f2c (ref. 12). We chose the f2c package because it is an open-source program and has been widely used in academic areas.

The f2c program translates Fortran 77 codes to C codes. Since our goal is to wrap the Fortran codes and provide the interface for both the client and the server, we only need to use f2c to extract and translate the codes of data types, variable declarations, and function headings. The function bodies and statements are of no interest to us. However, the codes converted by f2c do not totally meet the IDL syntactic requirements. For example, IDL requires a tag in the structure type, whereas C does not. Unfortunately, f2c translates a Fortran COMMON block into a structure variable in C without a tag. Furthermore, the structure tag can be used as a type to define structure variables in IDL. In C, it has to put the keyword `struct` in front of the tag, and this kind of declaration is not allowed in IDL.

The syntactic difference between the C and IDL data structure required some manual editing of the code generated from f2c in our work. This inconvenience motivated us to modify the f2c program by adding a few codes to generate the tag for a structure. Figure 2 shows an example of Fortran codes with the declarations of the COMMON blocks `cgcon` and `disp`. The corresponding codes in IDL, as translated by the modified f2c program (f2CORBA), are shown in figure 3.

Like the example shown in figure 2, most Fortran applications have several COMMON blocks. After decomposing the application into a few CORBA objects, the problem of passing the structure variables (i.e., COMMON blocks) to several servers needs to be solved. Our current approach is to merge all the structures into another structure (e.g., lu_tag in fig. 3) and use an attribute (e.g., lu_all in fig. 3) to facilitate data transfer.

This approach is based on the assumption that each server needs to access all the COMMON blocks. However, some servers may access only parts of them. A graphical user interface (GUI) to ease the conversion task has been partially developed. Programmers will be able to simply click on an item to select a structure variable from a list box to be a member in a structure-type attribute (see fig. 4). To implement the GUI interface, we are using the tool Tcl/Tk (ref. 13) because of its availability and portability. Tcl/Tk can be downloaded from the World Wide Web (ref. 14) and has been used on most operating system platforms, including UNIX, Windows NT (Microsoft Corp.), and Macintosh (Apple Corp.). Furthermore, most programmers can learn the fundamentals of Tcl/Tk and write script programs to do real work in a few days.

We have successfully tested the proposed conversion methodology on different CORBA packages: VisiBroker C++ (Inprise/Borland Corp., ref. 15) and MICO (ref. 16). Because of availability and portability, we prefer using MICO rather than VisiBroker C++. For example, VisiBroker C++, a commerical software program, only works with a Sparcworks C++ (Sun Microsystems) compiler on Sun Sparc or Ultra (Sun Microsystems) platforms. MICO, a public-domain ORB with a complete CORBA compliant implementation, relies on the GNU package and, hence, can be ported easily to almost any platform, including Solaris, LINUX, and Windows NT.

PERFORMANCE MEASUREMENTS

To investigate the overhead produced in distributed object computing, we selected the LU and BT benchmarks from the NAS Parallel Benchmarks (NPB) suite (ref. 17). These benchmarks, which were devised by the Numerical Aerodynamics Simulation (NAS) program at the NASA Ames Research Center, have been used widely to study the performance of parallel computing. For example, we used the benchmarks to evaluate the performance of a cluster of 32 Intel P6 (Intel Corp.) workstations that were connected by a two-level tree-structure network (ref. 18). The NPB 2.3 benchmarks are a set of eight problems consisting of five kernels that highlight specific areas of machine performance and three pseudoapplications that simulate computational fluid dynamics (CFD). Brief descriptions of the LU and BT benchmarks follow:

- **Application LU** solves a finite difference discretization of the three-dimensional compressible Navier-Stokes equations by using a symmetric successive over-relaxation (SSOR) numerical scheme.

- **Application BT** is based on a Beam-Warming approximate factorization that decouples the x, y, and z dimensions, resulting in three sets of narrow-banded, regularly structured systems of linear equations.

We used the sample-size serial version of the LU and BT benchmarks (NPB2.3-serial) and decomposed each benchmark into two server objects. The client needed to contact these two servers one after the other to accomplish the task. The experiments were performed on a pair of Sun Ultra computers (170–MHz, 128–MB) running Solaris 2.6 (Sun Microsystems) and also on a pair of Intel P6 computers (400-MHz, 512-MB) running LINUX kernel 2.2.12. The clients and servers were connected through a 100BaseT local area network (LAN).

Tables I and II show the breakdown of the elapsed time for running the benchmarks LU and BT, respectively. For comparison, we also ran the original programs. The results show that the time for service binding is small. However, the communication overhead, including the time for marshaling and unmarshaling data between the client and the servers, cannot be ignored.

CONCLUSION

Many scientific applications in aerodynamics and solid mechanics are written in Fortran. Refitting these legacy Fortran codes with Common Object Request Broker Architecture (CORBA) objects can increase the code reusability. In this paper, we have presented a methodology to integrate Fortran legacy programs into a distributed object framework. Issues and strategies regarding the conversion and decomposition of Fortran codes into CORBA objects have been discussed. We also have implemented a conversion tool that takes the Fortran application program as input and generates the C/C++ header file and the Interface Definition Language (IDL) file. The tedious programming tasks for wrapping the codes can, therefore, be reduced. In the future, we plan to add more user-friendly graphical user interfaces and to provide an analyzer tool to help programmers easily extract objects from legacy applications.

REFERENCES

1. Object Management Group: The Common Object Request Broker: Architecture and Specification, 2.3 edition, June 1999.
2. Vinoski, S.: CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Commun., vol. 35, Feb. 1997, pp. 46–55.
3. Haggerty, P.; and Seetharaman, K.: The Benefits of CORBA-Based Network Management. Communications of the ACM, vol. 41, Oct. 1998, pp. 73–79.
4. Pavon, J., et al.: CORBA for Network and Service Management in the TINA Framework. IEEE Commun., vol. 36, Mar. 1998, pp. 72–79.
5. Forslund, J., et al.: TeleMed: Development of a Java/CORBA-based Virtual Electronic Medical Record. Proceedings of the PacMedTek Symposium, Aug. 1998.
6. Sun Microsystems, Inc.: *Distributed Object Technology in the Financial Services Industry, White Paper*, last modified 1995. http://www.sun.com/software/solutions/third-party/software/whitepapers/ Accessed May 18, 2000.
7. *OMG*, last modified 2000. http://www.omg.org Accessed after June 1999.
8. Lytle. J.: The Numerical Propulsion System Simulation: A Multidisciplinary Design System for Aerospace Vehicles, NASA/TM—1999-209194, 1999. http://gltrs.grc.nasa.gov/cgi-bin/GLTRS/browse.pl?/1999/TM-1999-209194.html (Also, Proceedings of the 14th International Symposium on Air Breathing Engines sponsored by the International Society for Air Breathing Engines, Sep. 1999.)
9. Sandia National Lab (Summers, R.M.; Peery, J.S.; Hogan, R.E.; Holmes, V.P.; and Miller, D.J.): *Coupling Finite Element Codes Using CORBA-Based Environments*, last modified Jan. 4, 1996. http://www.cs.sandia.gov/HPCCIT/corba/impres.html Accessed May 18, 2000.
10. Achee, B.L.; and Carver, D.L.: Creating Object-Oriented Designs From Legacy Fortran Code. J. Syst. Software, vol. 39, 1997, pp. 179–194.
11. Ong, C.; and Tsai, T.: Class and Object Extraction from Imperative Code. J. Object-Oriented Prog., Mar./Apr. 1993, pp. 58–68.
12. Feldman, S.I., et al.: A Fortran-to-C Converter. Technical Report No. 149, Bell Laboratories, NJ, 1995.
13. Ousterhout, J.: Tcl and the Tk Toolkit. Addison-Wesley, Reading, MA, 1994.
14. Tcl/Tk 8.3.1 Download, last modified March 3, 2000. http://dev.scriptics.com/software/tcltk/download83.html Accessed May 9, 2000.
15. Inprise, Corp.: VisiBroker for C++: Programmer's Guide, Version 3.3, 1999.
16. Römer, K.; Puder, A.; and Pilhofer, F.: MICO is CORBA, last modified Feb. 9, 1997. http://www.mico.org Accessed May 18, 2000.
17. Bailey, D., et al.: The NAS Parallel Benchmarks 2.0. Technical report, NAS–95–020, NASA Ames Research Center, 1995.
18. Sang, J., et al.: High-Performance Cluster Computing Over Gigabit/Fast Ethernet. Informatica, vol. 23, 1999.

TABLE I.—BREAKDOWN OF ELAPSED TIME FOR RUNNING THE CLIENT-SERVER BENCHMARK LU

Benchmark LU	Elapsed time, sec					
	Traditional		One client, two servers			
	Computation	Total	Binding	Computation	Communication	Total
Sun Ultra	2.99	3.05	0.046	2.99	0.33	3.37
PC LINUX	1.51	1.58	.019	1.51	.19	1.72

TABLE II.—BREAKDOWN OF ELAPSED TIME FOR RUNNING THE CLIENT-SERVER BENCHMARK BT

Benchmark BT	Elapsed time, sec					
	Traditional		One client, two servers			
	Computation	Total	Binding	Computation	Communication	Total
Sun Ultra	6.99	7.25	0.047	6.99	2.12	9.16
PC LINUX	4.16	4.35	.019	4.16	1.38	5.73

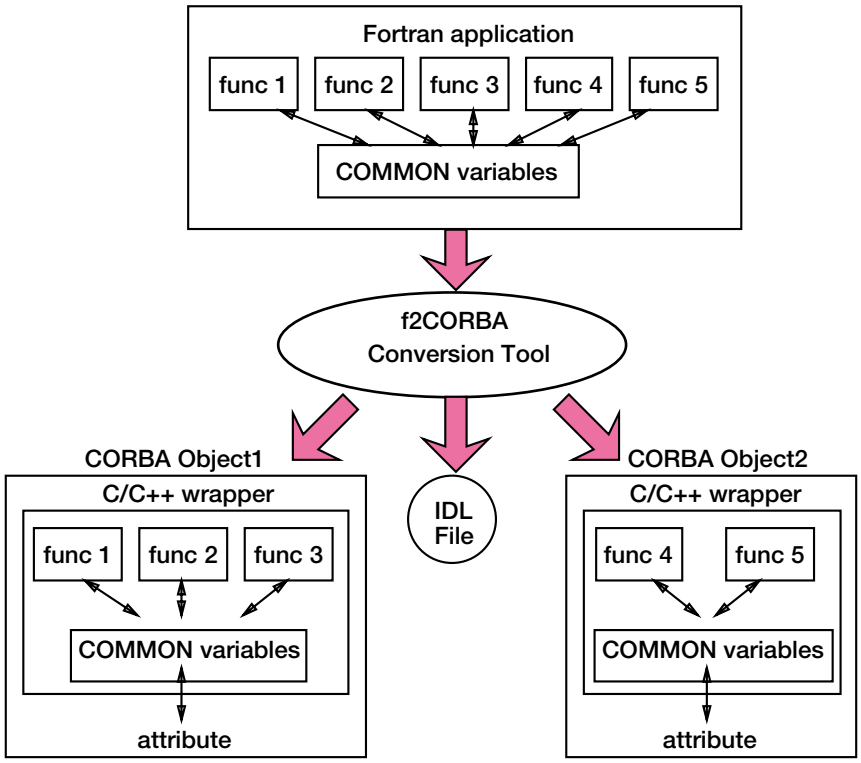

Figure 1.—Functional diagram of f2CORBA conversion tool. CORBA, Common Object Request Broker Architecture; IDL, Interface Definition Language.

```
integer nx, ny, nz
 integer nx0, ny0, nz0
 ...
 double precision dxi, deta, dzeta
 double precision tx1, tx2, tx3
 double precision ty1, ty2, ty3
 ...
 common/cgcon/ dxi, deta, dzeta,
>              tx1, tx2, tx3,
>              ty1, ty2, ty3, ...
>              nx, ny, nz,
>              nx0, ny0, nz0,
 ...
 double precision dx1, dx2, dx3, dx4, dx5
 double precision dy1, dy2, dy3, dy4, dy5
 ...
 common/disp/ dx1, dx2, dx3, dx4, dx5,
>             dy1, dy2, dy3, dy4, dy5,
 ...
```

Figure 2.—Original Fortran codes with COMMON block variables.

```
struct cgcon_tag {
     double dxi, deta, dzeta, tx1, tx2, tx3, ty1, ty2, ty3, ... ;
     integer nx, ny, nz, nx0, ny0, nz0, ... ;
} ;
struct disp_tag {
     double dx1, dx2, dx3, dx4, dx5, dy1, dy2, dy3, dy4, dy5, ... ;
} ;
...
struct lu_tag {
     cgcon_tag cgcon_;
     disp_tag disp_;
...
} ;
interface Lu1 {
     attribute lu_tag lu_all;
     void lu1_comp();
} ;
interface Lu2 {
     attribute lu_tag lu_all;
     void lu2_comp();
} ;
```

Figure 3.—Converted codes in Interface Definition Language (IDL).

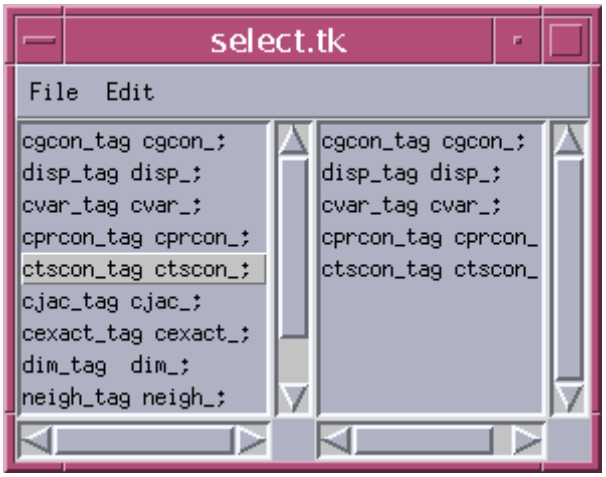

Figure 4.—Graphical user interface for structure variable selection.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 2000	Technical Memorandum

4. TITLE AND SUBTITLE

Developing CORBA-Based Distributed Scientific Applications From Legacy Fortran Programs

5. FUNDING NUMBERS

WU–509–10–24–00

6. AUTHOR(S)

Janche Sang, Chan Kim, and Isaac Lopez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–12204

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
Cleveland, Ohio 44135–3191
and
NASA Glenn Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—2000-209950
ARL–MR–488

11. SUPPLEMENTARY NOTES

Prepared for the Computational Aerosciences Workshop sponsored by the High Performance Computing and Communications Program, Moffett Field, California, February 15–17, 2000. Janche Sang, Cleveland State University, Department of Computer and Information Science, Cleveland, Ohio 44115; Chan Kim, NASA Glenn Research Center; and Isaac Lopez, U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio 44135. Responsible person, Isaac Lopez, organization code 2900, (216) 433–5893.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 01 and 61 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

An efficient methodology is presented for integrating legacy applications written in Fortran into a distributed object framework. Issues and strategies regarding the conversion and decomposition of Fortran codes into Common Object Request Broker Architecture (COBRA) objects are discussed. Fortran codes are modified as little as possible as they are decomposed into modules and wrapped as objects. A new conversion tool takes the Fortran application as input and generates the C/C++ header file and Interface Definition Language (IDL) file. In addition, the performance of the client server computing is evaluated.

14. SUBJECT TERMS	15. NUMBER OF PAGES
CORBA; Fortran	13
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	